- Search user created libraries to satisfy unresolved global symbols
- Dynamically assign memory
- Create a memory map describing the location of each object module and data block loaded

PROGRAM SEGMENTATION

The Linking Loader and Macro Assembler permit the user to segment source programs into five different sections. These sections and their corresponding functions are as follows:

ASCT - Absolute Section (non-relocatable)

There may be a limited number of absolute sections in a user's program. These sections are used to allocate/load/initialize memory locations assigned by the programmer rather than the loader, for example, addresses assigned to ACIA's and PIA's.

BSCT - Base Section

There is only one Base Section. The linking loader allocates portions of this section to each module that needs space in BSCT. BSCT is generally used for variables that will be referenced via direct addressing. BSCT is limited to locations 0-255 of the addressing range.

CSCT - Blank Common (uninitialized)

There is only one CSCT. This section is used for blank common (similar to FORTRAN blank common). This section cannot be initialized.

- DSCT - Data Section

  There is only one Data Section. The linking loader allocates portions of this section to each module that needs a part of DSCT. DSCT is generally used for variables (RAM) which are to be accessed via extended mode addressing.

- PSCT - Program Section

  PSCT is similar to DSCT except that it is intended to be used for instructions. The PSCT/DSCT division was made to facilitate a RAM/ROM dichotomy.

This section concept is preserved by the Loader during the load process. As a module is being loaded, each of its sections is combined with the corresponding sections of previously loaded modules. As a result, the absolute load module produced by the Loader will contain one continuous memory area for each section type encountered during the load operation.

In addition to the program segmentation provided by the section concept, the M6800 relocation and linking scheme supports named common. The named common concept provides the function of initializable common areas within BSCT, DSCT, and PSCT. In processing named common definitions, the Loader shall:

- Assign to each named common area a size equal to the largest size defined for the named common during the load process.
- Allocate memory at the end of each section for the named common blocks defined within that section.

The load maps shown in Figure I-1 describe the load process with regard to sections and named common. The module PGM1 requires memory to be reserved in BSCT, CSCT, DSCT, and PSCT, although the only space necessary in DSCT is for the named common NCOM1. The module PGM2 requires that memory be allocated in BSCT, CSCT, DSCT, and PSCT. Neither module defines any ASCT blocks.

The load module's map illustrates a typical memory map that might be produced by loading PGM1 and PGM2. The BSCT for both PGM1 and PGM2 are allocated memory within the first 256 bytes of memory. As shown, the first 32 (20 hex) bytes of BSCT are reserved by the Loader for use by the disc operating system unless otherwise directed. After BSCT, space for blank common is allocated, followed by space for PGM2's DSCT. Since PGM1 requires no DSCT for its exclusive use, none will be allocated. The named common block NCOM1 within DSCT is assigned memory at the end of DSCT. Finally, the PSCT's for PGM1 and PGM2 are allocated along with PSCT's common blocks NCOM2 and NCOM3.

The Loader assigns memory within sections in the order in which the modules are specified. Named common blocks are allocated memory at the end of their corresponding section, in the order in which they are defined. Figure I-2 illustrates a load module map produced by loading PGM2, followed by PGM1. This load module map is slightly different from the map in Figure I-1 where PGM1 was loaded first.

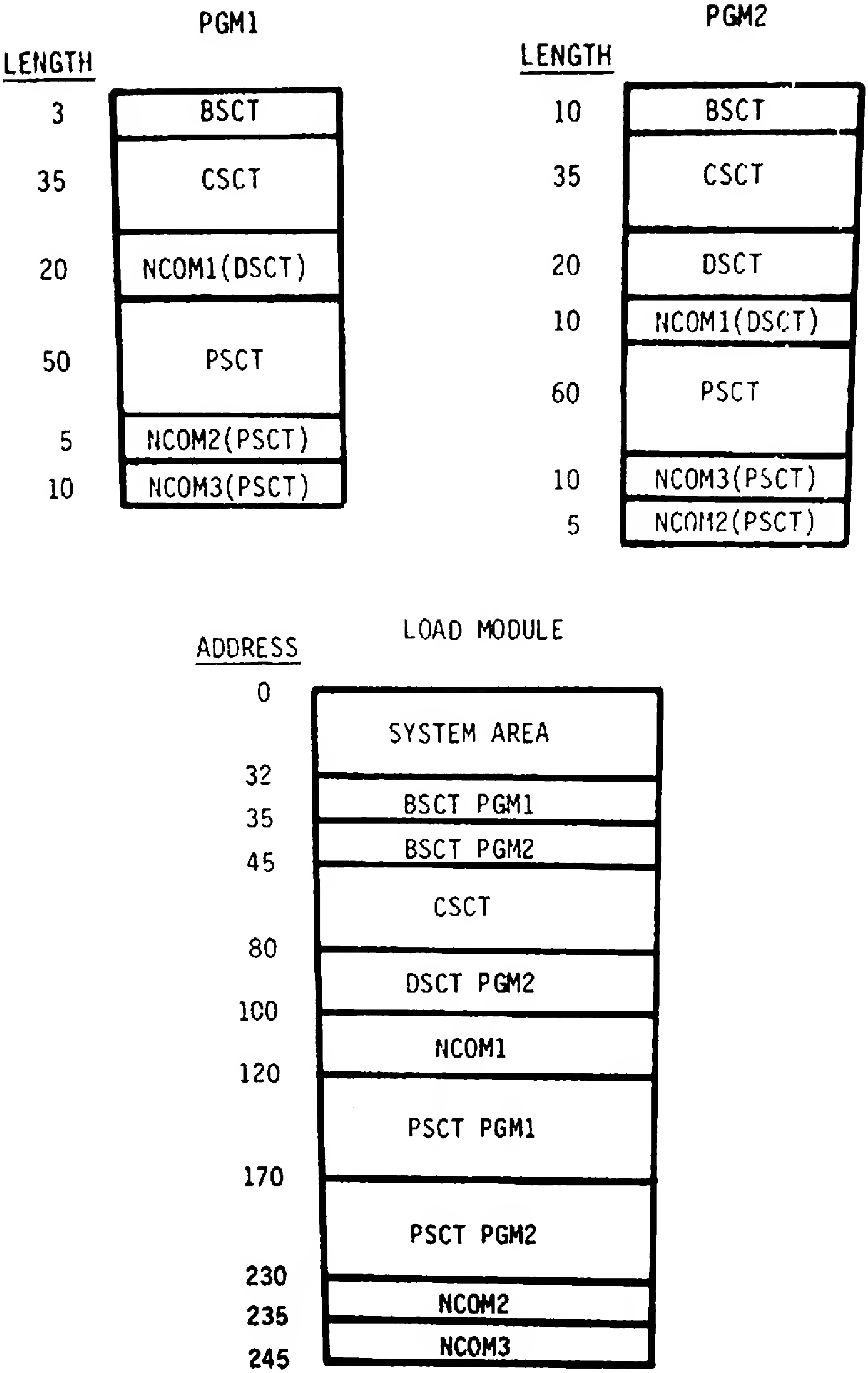


FIGURE I-1

LOAD MAPS

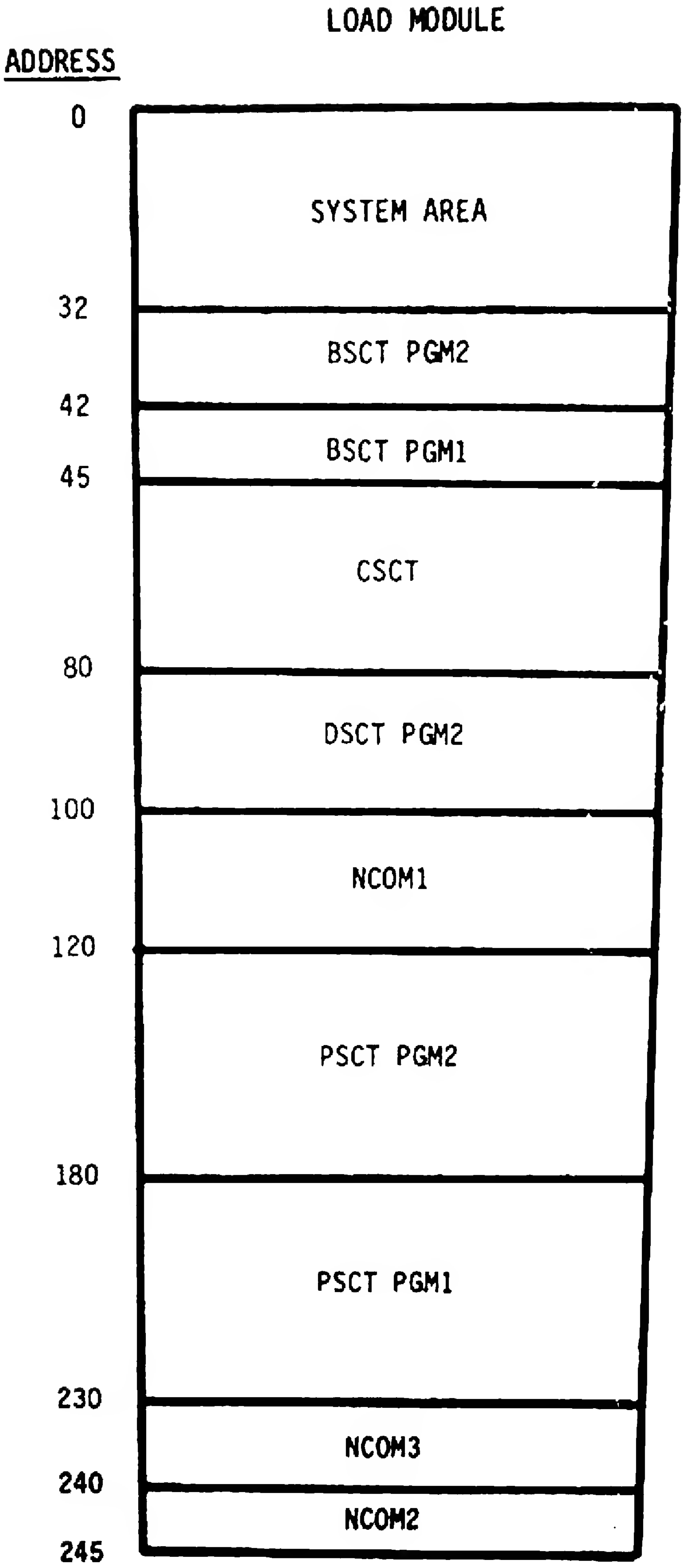


FIGURE I-2

LOAD MAP

RELOCATION

Relocation allows the user to assemble a source program without assigning absolute addresses at assembly time. Instead, absolute memory assignment is performed at load time. In order to relocate a program (within memory), the source program must be assembled with the M6800 Macro Assembler using the OPT REL directive. Programs assembled with this directive will cause the assembler to produce a relocatable object module instead of an absolute object module. These relocatable object modules contain information describing the size of each section (ASCT, BSCT, CSCT, and DSCT) and named common area as well as the relocation data. A complete description of the relocatable object module format is contained in the M6800 Macro Assembler Manual.

In order to load a relocatable object module, the M6800 Linking Loader must be used. The Loader assigns load addresses and produces an absolute object module compatible with the EXORciser loader.

The advantages of using relocation are:

- Reassembly is not required for each new absolute load address.
- Relocation via the M6800 Linking Loader is faster than reassembly.
- Dynamic memory assignment of modules is possible.

LINKING

Linking allows instructions in one program to refer to instructions or data which reside within other programs. If all programs are assigned absolute addresses during assembly time, it is possible to directly reference another program via absolute addresses. However, when using relocatable programs, absolute load addresses are not generally known until load time. In order to access other relocatable programs or data blocks, external reference symbols must be used. These external symbols are commonly called global symbols since they may be referenced by any module at load time. Although global symbols are used to link modules at load time, they must be explicitly defined and referenced at assembly time. This is accomplished by the M6800 Macro Assembler directives, XDEF and XREF. The XDEF directive indicates which symbols defined within a module can be referenced by other modules. The XREF directive indicates that the symbol being referenced is defined outside the module.

At load time, global references are matched with their corresponding global definitions. Any reference within a module to a global symbol is updated with the load address of the global symbol. If the loader detects a global reference without an associated global definition, an undefined global error will be printed and a load address of zero will be assigned to the reference.

MODULE LIBRARIES

The M6800 Linking Loader can automatically search a file for modules which contain definitions satisfying any unresolved global symbols. Such a file is called a library file and is composed of one or more object modules. The Loader sequentially searches the library file. If a module is found which contains a symbol definition satisfying an unresolved global symbol, the module will be loaded. Only those modules which can satisfy an unresolved reference will be loaded. Since a library file is searched only once, modules which reference other modules within the library file should occur within the library file before the referenced module. Otherwise, the user must direct the Loader to search the library again.

MEMORY ASSIGNMENT

During the load process, absolute addresses are assigned to the program sections within the specified modules. Normally the loader will automatically perform this assignment by allocating memory by sections in the order: ASCT, BSCT, CSCT, DSCT and PSCT. However, the user may define the starting and/or ending address of any non-ASCT section. In this case, the Loader will first reserve memory for those sections with defined load addresses before allocating space for any other section. The Loader also permits a user to specify the relative section offset of a module within a section. However, a section of a module is always loaded in the associated load section in the order in which the module was specified.

LOAD MAPS

The Loader will optionally produce a load map describing the memory layout resulting from the load of the specified modules. Figure I-3 is an example of some of the features included in a typical load map. In addition to this full load map, the Loader may be directed to produce partial load maps listing only the undefined global symbols or section load addresses.

OPERATING ENVIRONMENT

Equipment Requirements

Minimum equipment requirements for the M6800 Linking Loader include:

- EXORciser
- 10K bytes of RAM
- Floppy Disc
- Console

Software Requirements

The M6800 Linking Loader operates under the EDOS2.3 floppy disc operating system to load relocatable object modules produced by the M6800 Macro Assembler.

SECTION II

USING THE M6800 LINKING LOADER

CALLING THE LINKING LOADER

The M6800 Linking Loader must be called while under the control of the disc operating system. When the user types the command

```
        RLOAD    < c/r >
```

the disc executive will load the Linking Loader. Upon entry, the loader prints

```
        M6800 LINKING LOADER REV n.m
        ?
                                    (where n.m is the revision number)
```

The character '?' is the Loader's prompt and is printed whenever the Loader has completed the last command and is ready for another.

LOADER INPUT

The input to the Loader is in one of two forms - commands and object modules. The Loader commands control the relocation and linking of desired object modules. The object modules are produced by the M6800 Macro Assembler when the relocation option is specified. Each source program assembled by the Macro Assembler creates a single relocatable object module on a disc file. These disc files or those files created by merging one or more of these files are used as the input to the Loader.

The Loader command structure provides for the loading of an entire file or selected modules within a file. In addition, a disc file may be used as a library file.

COMMAND FORMAT

Each Loader command line consists of a sequence of commands and comments followed by a carriage return. The first blank in a command line terminates the command portion of the line and the remainder is assumed to be comments. Multiple commands may appear on a line by using a semi-colon (;) as a command separator. The format of a command line may thus be defined as:

$$\left[<\text{command}> \left[;<\text{command}> \right]_{0}^{00} \right] \left[<\text{space}> \left[<\text{comments}> \right] \right] <\text{c/r}>$$

The commands in a command line are executed only after the Loader detects a carriage return.

If a command line is entered incorrectly, the line may be corrected in either of two manners. First, the command line may be deleted completely by typing CTRL X (the CTRL and X keys typed simultaneously). This causes the Loader to ignore the current command line and a new prompt (?) will be printed. Instead of deleting the entire command line, the command line may be corrected by deleting the character(s) in error. This is accomplished by typing a RUBOUT to delete the last character typed. The typing of a RUBOUT also causes the last character to be printed. After deleting the character(s) in error, the corrected version of the command line may be entered.

The Loader will execute all the commands in a command line before another prompt is issued. If an error is detected while attempting to process a command, that command will be terminated. The remaining commands in the command line will be ignored.

When using multiple commands per line, it should be noted that selected commands require that they are the last command on a line. These commands include:

- INIT
- All intermediate file commands (IF, IFON, IFOF)
- ABSP when used in conjunction with an intermediate file

LOADER COMMANDS

The Loader commands are divided into three classes: (1) control commands; (2) load directives; (3) state directives. The control commands are used to initiate Pass I and II of the Loader as well as to return to EXBUG or the disc operating system. The load directives are used to identify the modules to be loaded. Finally, the state directives direct the assignment of memory to the various program sections and the production of a load map.

Command Nomenclature

<f_name> - Used to indicate the name of a disc input file. Disc file names must start with an alphabetic character and contain a maximum of five alphanumeric characters. The disc unit upon which the file resides may be defined by placing a colon (:) after the file name, followed by a valid disc drive number. If a disc unit is not specified, drive zero will be assumed.

<m_name> - Used to indicate a named module. Named modules are composed of a maximum of six alphanumeric characters, the first of which must be alphabetic.

<name> - Used to indicate a named file or module as determined by the Loader's current file/module state.

<number> - Used to indicate a decimal or hexadecimal number. Unless preceded by a '$' character which is used to denote hexadecimal, the number will be interpreted as decimal. The allowable number range unless explicitly stated otherwise will be:

0 - 65,535 (decimal)
0 - $FFFF (hexadecimal)

[ ] - Used to indicate that the enclosed directive(s) is optional.

$[\quad]_{0}^{00}$ - Used to indicate that the enclosed directive may be repeated any number of times.

{ } - Indicates that one of the enclosed options <u>must</u> be used.

Control Commands

ABSP - Produce Absolute Load Module

FORMAT: ABSP [=<m_name>[,<printable information>]]

DESCRIPTION: ABSP initiates the second pass of the Loader. During this pass, an absolute binary memory image is produced in EXORciser loadable format on the disc file defined by the BO command. If an output module name is specified, it will be included in the module's S0 record. Any printable information is also included in the S0 record if specified. The printable information may contain any character and is terminated only by a semi-colon or carriage return. NOTE: A space is a valid character in the printable information and does not terminate the command line. The module name and printable information may not exceed 30 characters.

If an intermediate file (IF) was generated during Pass I, the second pass of the Loader will proceed automatically as directed by the commands entered during the first pass. When an IF is being used, the ABSP command must be the last command in a command line.

In the event that an IF is not created during Pass I, the same sequence of commands used in Pass I (with the exception of the MAP commands) must be repeated exactly as in Pass I.

Prior to the ABSP command, a binary output file must be defined via the BO command.

EXAMPLE: ABSP=ROOT, A SQUARE ROOT PROGRAM

As a result of this command, the second pass of the assembler is initiated to produce an absolute module. The phrase 'ROOT, A SQUARE ROOT PROGRAM' is written in the S0 record of the absolute module.

BO - Binary Output

FORMAT: BO = < f_name >

DESCRIPTION: The BO command is used to direct the binary output in EXORciser load format to a disc file. The disc file defined by the BO command must not currently exist on the defined drive.

EXAMPLE:	BO=BOBJ	Write binary load module on file BOBJ on drive 0
	BO=BOBJ1:1	Write binary load module on file BOBJ1 on drive 1

EXBUG

FORMAT: EXBUG

DESCRIPTION: The EXBUG command is one of two commands which may be used to exit the Loader. EXBUG causes control to be returned to the EXORciser's EXBUG mode after all Loader files are closed.

EXIT

FORMAT: EXIT

DESCRIPTION: The EXIT command is one of two commands which terminates the Loader's activity. EXIT causes control to be returned to the disc operating system.

IDOF - Suppress Printing of Module ID

FORMAT: IDOF

DESCRIPTION: The IDOF command suppresses the printing of the module name and print information associated with each object module loaded. The Loader is initialized to the IDOF state.

IDON - Print Module ID

FORMAT: IDON

DESCRIPTION: This command causes the printing on the console of the name and printable information associated with each object module loaded or encountered in a library file.

IF - Intermediate File

FORMAT: IF = < f_name >

DESCRIPTION: The IF command defines a file to be used as an intermediate file. An intermediate file is a copy of all Pass I Loader commands and object modules. It is used to direct the Loader durirg Pass II, instead of requiring the user to retype the Pass I command sequence during Pass II. The IF command also automatically places the Loader in intermediate file mode similar to the IFON command. Like the IFON command, the IF command <u>must</u> be the last command in a command line.

The IF file name must be a valid disc file name and may not be the name of an existing file on the specified disc unit.

EXAMPLE: IF=IFILE   Defines IFILE on drive 0 as the intermediate file.

IFOF - Intermediate File Mode Off

FORMAT: IFOF

DESCRIPTION: IFOF temporarily suppresses the creation of the intermediate file until an IFON directive is encountered. This command must be the last command in a command line.

IFON - Intermediate File Mode On

FORMAT: IFON

DESCRIPTION: This command directs the Loader to write all further commands and object modules onto the intermediate file. This directive remains in effect until an IFOF or Pass II command is detected. The IFON command must be the last command on a command line. IFON is implied when the intermediate file is defined by the IF command. If an intermediate file is to be used during Pass II, the IFON directive must be in effect.

INIT - Initialize Loader

FORMAT: INIT

DESCRIPTION: INIT initializes the Loader for Pass I. This command is performed automatically when the Loader is first initiated. The use of this command permits several output object modules to be created by the Loader. The INIT command _must_ be the last command in a command line.

OI - Object Input

FORMAT: OI = < f_name >

DESCRIPTION: The OI command is used to identify an input file containing one or more object modules. The file name must be the name of an existing disc file.

EXAMPLE:	OI=PGM1	Object input on file PGM1 on drive 0
	OI=PGM2:0	Object input on file PGM2 on drive 0

Load Directives

FILE - File Mode

FORMAT: FILE

DESCRIPTION: The FILE directive is used to place the Loader in file mode. While in file mode, the Loader will operate on all the modules within a file as directed by the load directives. The file mode is the default mode. The file mode may be temporarily overridden by the ':M' option of the LOAD command.

LIB - Library Search

FORMAT: $$\text{LIB}\ \Bigg[\Big[=\begin{matrix}<\text{number}>\\<\text{name}>\end{matrix}\Big]\Big[:\text{M}\Big]\Bigg]\Bigg[,\Big[\begin{matrix}<\text{number}>\\<\text{name}>\end{matrix}\Big]\Big[:\text{M}\Big]\Bigg]_{0}^{00}\Bigg]$$

DESCRIPTION: The LIB command instructs the Loader to search the specified object modules for those modules which satisfy any undefined global references. Any module that satisfies a global symbol will be loaded.

The object modules to be searched are specified in the same manner as explained in the description of the LOAD command.

Modules loaded via the LIB command may also reference global symbols that are not defined. Since a library file is searched once for each LIB command, care should be taken when creating a library file in order to avoid multiple passes of the same library file.

EXAMPLE:	LIB	Searches the remaining modules on the input file to resolve unsatisfied global references
	LIB=MLIB:1	The file MLIB on disc unit 1 will be used as a library file.

LOAD - Load a File or Module

FORMAT: $$\mathrm{LOAD}\ \left[\left[=\begin{matrix}<\text{number}>\\<\text{name}>\end{matrix}\right]\left[:M\right]\left[,\left[\begin{matrix}<\text{number}>\\<\text{name}>\end{matrix}\right]\left[:M\right]\right]_{0}^{\infty}\right]$$

DESCRIPTION: The LOAD command directs the M6800 Linking Loader to load the specified object files and/or modules.

If a < number > is given, the Loader will load the next < number > of modules from the disc file defined by the OI command. When the < number > format of the LOAD command is used, the ':M' feature or the MODU directive must be in effect. The ':M' option causes the Loader to enter module mode only for the indicated sub-command. A maximum of 255 modules may be loaded at one time with this form.

The use of the < name > form of the LOAD command causes the Loader to load the defined module or file. The < name > must be a valid file or module name. To load a module by name, the ':M' feature or the MODU directive must be in effect and the module must be contained within the disc file defined by the OI command.

NOTE: Disc files are sequential files and are not rewound prior to a module search.

When no options are specified as part of the LOAD command, only one file or module will be loaded from the disc file defined by the OI command.

EXAMPLE: LOAD=PGM1:1 — Loads all modules within file PGM1 on disc drive 1

LOAD=1:M,PGM2:M — Loads from the input file the next module and the module named PGM2

LOAD=PGM3 — Loads the file PGM3 from drive Ø or the module PGM3 from the defined input file. The file/module mode of the Loader determines whether a file or module will be loaded.

MODU - Module Mode

FORMAT: MODU

DESCRIPTION: The MODU directive places the Loader in the module mode. While in the module mode, the < name > and < number > options of the load directives shall refer to modules.

SKIP - Skip Input Modules

FORMAT: SKIP=<number>[:M]

DESCRIPTION: The SKIP command directs the Loader to skip the defined number of modules in the file indicated by the OI command. The MODU directive or ':M' option must be in effect. A maximum of 255 modules may be skipped with a single command.

EXAMPLE: SKIP=2:M Skips the next two modules on the input file

SRCH - Search for a File or Module

FORMAT: SRCH=< name > [:M]

DESCRIPTION: SRCH causes the Loader to search for a named object module or file. If the MODU directive is in effect or the ':M' option specified, the current disc file defined by the OI command will be searched for the named module. If the Loader is operating in the file mode as directed by the FILE command, a disc search will be performed for the named file. If the named file is found, this file will become the new object input file for future Loader commands. When in file mode, the file named must be a valid file name and the drive unit may be given by typing a colon (:) and the drive number after the file name. If no drive unit is specified, drive 0 is assumed.

EXAMPLE: SRCH=FADD:1 Searches disc unit 1 for the file FADD. If found, FADD will be the new input file.

SRCH=SINE:M Searches for the module named SINE on the current input file.

State Commands

CUR - Set Current Location Counter

FORMAT: CUR $\left\{\begin{matrix}B\\D\\P\end{matrix}\right\}$ $\left[\backslash\right]$ < number >

DESCRIPTION: The CUR command causes the Loader to set the current relative address of the specified section (BSCT, DSCT, or PSCT) to the given number. The defined number must be greater than or equal to the section's current location counter address. The '\' option causes the Loader to start the specified section of all future modules loaded on an address modulo the given number. The '\' option remains in effect until revoked with a '\Ø' option or until the current pass of the Loader is complete. If the '\' option is in effect when memory is assigned, the start address of the section will be modulo the given number. The '\' option does not apply to named common blocks within the specified section.

EXAMPLE: `CURP=$100` Sets the relative PSCT location counter to 100 (hexadecimal).

`CURP=\$100` Causes the Loader to update PSCT's relative location counter to the next modulo 100 (hexadecimal) address. This function is performed for each module loaded after this command.

DEF - Loader Symbol Definition

FORMAT:

$$\text{DEF:}\langle\text{name1}\rangle = \left\{\begin{matrix}\langle\text{number}\rangle\\ \langle\text{name2}\rangle\end{matrix}\right\}\left[\,,\begin{matrix}\text{ASCT}\\ \text{BSCT}\\ \text{DSCT}\\ \text{PSCT}\end{matrix}\right]$$

DESCRIPTION: The DEF command is used to define a global symbol and enter it in the global symbol table. The symbol to be defined is given by name1 and must be a valid Macro Assembler variable name. The symbol may not currently be defined. If the <number> option is used, the symbol will be defined with the given number as the relative address within the specified section. The DEF command may be used to provide another name for a previously defined symbol by using the <name2> option. <name2> must be a currently defined global symbol. The section options - ASCT, BSCT, DSCT, PSCT - are used to define the section associated with the defined section. ASCT is the default section.

EXAMPLE: `DEF:ACIA1=$EC10,ASCT` Defines symbol ACIA1 as an ASCT symbol with absolute address EC10 (hexadecimal).

END - Ending Address

FORMAT: $\text{END}\left\{\begin{matrix}\text{B}\\ \text{C}\\ \text{D}\\ \text{P}\end{matrix}\right\} = <\text{number}>$

DESCRIPTION: The END commands are used to set the absolute ending address of the associated section (BSCT, CSCT, DSCT or PSCT). If both an ending and starting address are defined, the size described by these boundaries must be greater than or equal to the size of the associated section.

EXAMPLE: ENDB=255 BSCT will be allocated such that the last address reserved is 255 (decimal).

MAP - Prints Load Maps

FORMAT: MAP $\left\{\begin{matrix} C \\ F \\ S \\ U \end{matrix}\right\}$

DESCRIPTION: The MAP commands are used to display the current state of the modules loaded or the Loader's state directives.

MAPC - Prints the current size, user defined starting address, and user defined ending address for each of the sections, as well as the size, starting address, and ending address for each ASCT defined.

MAPF - A full map of the state of the loaded modules is produced after the Loader assigns memory. This map includes a list of any undefined symbols, a section load map, a load map for each defined module and named common and a defined global symbol map.

MAPS - The Loader assigns memory to those sections not defined by a user supplied starting and/or ending address. A memory load map which defines the size, starting address and ending address for each section is printed.

MAPU - Prints a list of all global references which currently remain undefined.

STR - Starting Address

FORMAT: STR $\begin{Bmatrix} B \\ C \\ D \\ P \end{Bmatrix}$ = < number >

DESCRIPTION: The STR commands set the absolute starting address of the associated section (BSCT, CSCT, DSCT, PSCT). Those sections whose starting address is not defined by the user will be assigned a starting address by the Loader.

EXAMPLE:

STRP=\$1000	PSCT will be allocated memory starting at 1000 (hexadecimal).
STRB=0	Overwrites the default starting address of BSCT.

APPENDIX A

A SUMMARY OF M6800 LINKING LOADER COMMANDS

COMMAND	FUNCTION
CONTROL COMMANDS	
ABSP[=<m_name>[,<printable information>]]	Initiates Pass II
BO=<f_name>	Specify the binary object file
EXBUG	Give control to EXBUG
EXIT	Give control to the disc operating system
IDOF	Suppress identification printing
IDON	Print module identification information
IF=<f_name>	Specify the intermediate file
IFOF	Intermediate file mode off
IFON	Intermediate file mode on
INIT	Initialize the Loader
OI=<f_name>	Specify the object input file
LOAD DIRECTIVES	
FILE	Enter file mode
LIB $\left[\left[=\begin{matrix}<\text{number}>\\<\text{name}>\end{matrix}\right][:M]\left[,\left[\begin{matrix}<\text{number}>\\<\text{name}>\end{matrix}\right][:M]\right]_0^{00}\right]$	Library search and load
LOAD $\left[\left[=\begin{matrix}<\text{number}>\\<\text{name}>\end{matrix}\right][:M]\left[,\left[\begin{matrix}<\text{number}>\\<\text{name}>\end{matrix}\right][:M]\right]_0^{00}\right]$	Load the indicated file(s)/module(s)
MODU	Enter module mode
SKIP=<number>[:M]	Skip files/modules
SRCH=<name>[:M]	Search for a file or module

COMMAND	FUNCTION
STATE COMMANDS	
CUR $\left\{\begin{matrix}B\\D\\P\end{matrix}\right\}$ = [\\] <number>	Set current location counter
DEF:<name1> = $\left\{\begin{matrix}\text{<number>}\\\text{<name2>}\end{matrix}\right\}$ $\left[\begin{matrix}\text{ASCT}\\\text{BSCT}\\\text{.DSCT}\\\text{PSCT}\end{matrix}\right]$	Define a symbol
END $\left\{\begin{matrix}B\\C\\D\\P\end{matrix}\right\}$ = <number>	Set section ending address
MAPC	List user assigned section sizes and addresses
MAPF	List full load map
MAPS	List loader assigned section sizes and addresses
MAPU	List undefined symbols
STP $\left\{\begin{matrix}B\\C\\D\\P\end{matrix}\right\}$ <number>	Set section starting address

APPENDIX B

LINKING LOADER ERROR MESSAGES

Errors detected by the Linking Loader while processing a command or loading a module will result in an error message being printed at the user's terminal. These errors are divided into two classifications: fatal errors and non-fatal (warning) errors. When the Loader detects a non-recoverable error, a fatal error message will be printed. Any commands not processed on the last command line will be ignored and a new prompt printed. If the Loader can recover from an error, only a warning message will be printed.

FATAL ERROR MESSAGES

Message	Explanation
BAE	BSCT Assignment Error - the combined size of BSCT is greater than the amount that can be allocated in the defined BSCT area.
COV	Common Overflow - the size of a section's common is greater than 65,536.
GAE	General Assignment Error - the Loader cannot assign absolute memory addresses. This may result from: • the definitions of ASCTs • user assignment of section addresses • the combined length of all sections exceeding 65,536 • the order in which the Loader assigns memory
ICM	Illegal Command

IOR Illegal Object Record - the input module is not a valid relocatable object module.

ISY Illegal Syntax - error in the option or specification field of a command. This error may also occur when a command is not terminated by a semi-color, space or carriage return.

LOV Local Symbol Table Overflow - not enough memory for all the global (external) symbols defined by the object modules.

SOV Section Overflow - the size of a section is greater than 65,535.

UAE User Assignment Error - the user has incorrectly defined load addresses. This error occurs when:

- the user defined end address is less than the user defined start address
- the space allocated by the user defined start and end addresses is less than that required for the section
- the user has defined load addresses which overlap.

UBO Undefined BO File

WARNING MESSAGES

Message Explanation

MDS-<symbol> Multiply Defined Symbol - the Loader has encountered another definition for the previously defined global <symbol>. Only the first definition will be valid.

UDS-<symbol> Undefined Symbol - the <symbol> was not defined during Pass I. A load address of zero will be assumed.

FATAL I/O ERRORS

If the Loader detects an error while attempting to read or write from disc, the following message will be printed.

DK xx ST yy f_name

where xx is the disc unit, yy is the I/O error, and f_name is the name of the file being referenced. If an I/O error is detected while creating an intermediate file, the Loader shall suspend the IF creation. In this event, the user should reinitialize the Loader.

I/O Errors	Explanation
3	Only one output file may be opened at any time
4	Device not ready
5	Invalid device (Loader error)
6	Duplicate file name
7	Named file does not exist
8	File not opened
9	Unexpected end-of-file
C	Directory or disc space full
E	Checksum error on object record

APPENDIX C

EXAMPLES OF LOAD OPERATION

This appendix serves to illustrate the major features provided by the M6800 Linking Loader. Figures C-1, C-2 and C-3 show three programs which have been assembled by the M6800 Macro Assembler. The relocatable object modules created by assembling these programs are used as input modules to the Loader in Figures C-4, C-5 and C-6. Figure C-4 illustrates the use of the Loader without an intermediate file. In Figure C-5, an intermediate file is created and an example of user defined starting addresses is shown. An illustration of library files is provided in Figure C-6. The library file PG120 was created by merging files PG10 and PG20.

```
PAGE  001   P01     PROGRAM TO PRINT OUT MESSAGES (MAIN)

00001                           NAM    P01
00002                           OPT    REL,CREF,NOG,LLEN=100
00003                           TTL    PROGRAM TO PRINT OUT MESSAGES (MAIN)
00004                           IDNT   MESSAGE PROGRAM 1
00005                      *
00006                      * COMMON MESSAGE AREA
00007                      *  (NAMED COMMON "BCOMM" IN BSCT)
00008                      *
00009N 0000                BCOMM  COMM   BSCT
00010N 0000    0000  P MSG1P  FDB    MSG1       PTR TO MESSAGE 1 (IN PSCT)
00011N 0002    0000  D MSG2P  FDB    MSG2       PTR TO MESSAGE 2 (IN DSCT)
00012N 0004    0000  A MSG3P  FDB    MSG3       PTR TO MESSAGE 3 (XREF IN BSCT)
00013N 0006    0000  A MSG4P  FDB    MSG4       PTR TO MESSAGE 4 (XREF IN DSCT)


00015                      *
00016                      * MESSAGES 1 AND 2
00017                      *  (NEW NAMED COMMON "BCOMM2" IN BSCT)
00018                      *
00019N 0000                BCOMM2 COMM   BSCT
00020N 0000    0001  A CMSGCT RMB    1          COMMON MESSAGE COUNT
00021N 0001    0014  A CMSG   RMB    20         COMMON MESSAGE


00023C 0000                       CSCT          BLANK COMMON SECTION
00024C 0000    0010  A MSGCST RMB    $10        RESERVE 16 BYTES


00026D 0000                       DSCT          DATA SECTION
00027D 0000    4D    A MSG2   FCC    /MESSAGE 2/
00028D 0009    04    A        FCB    4          DELINEATE END OF MESSAGE


00030P 0000                       PSCT          PROGRAM SECTION
00031P 0000    4D    A MSG1   FCC    /MESSAGE 1/
00032P 0009    04    A        FCB    4


00034                     *
00035                     * EXTERNAL REFERENCES
00036                     *
00037                             XREF   ATEST
00038                             XREF   BSCT MSG3,DSCT MSG4,ANY STACK,EXBENT,PGM2


00040                     *
00041                     * EXTERNAL DEFINITIONS
00042                     *
00043                             XDEF   MSG2,MSG1,EXBPRT,START,PGONE

00045           F024  A EXBPRT EQU    $F024    EXBUG PRINT ROUTINE
```

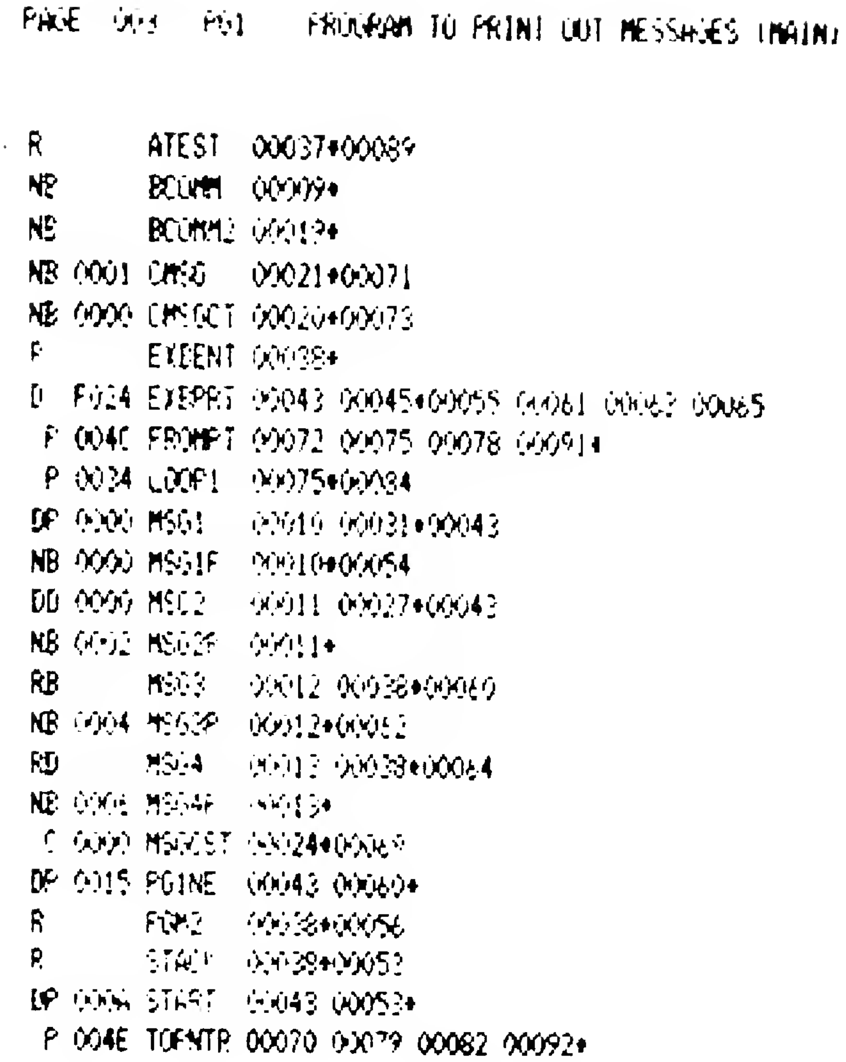
FIGURE C-1

MESSAGE PROGRAM 1

```
PAGE  002   PG1    PROGRAM TO PRINT OUT MESSAGES (MAIN)

00047                          *
00048                          * PROGRAM SECTION
00049                          * EXECUTION STARTS AT "START"
00050                          *
00051P 0000A                          PSCT          PROGRAM SECTION

00053P 0000 8E [illegible]  A START   LDS   #STACK  SET UP STACK REGISTER (XREF)
00054P [illegible] DE [illegible] N    LDX   MSG1P   GET MESSAGE 1 POINTER
00055P [illegible] BD [illegible] A    JSR   EXEPRT  PRINT MESSAGE 1
00056P [illegible] 7E [illegible] A    JMP   PGM2    GO TO PROGRAM 2 (XREF
00057                          *
00058                          * PROGRAM 2 RETURNS TO THIS POINT (XDEF)
00059                          *
00060P [illegible] CE [illegible] A POINE LDX   #MSG3   GET MESSAGE 3 ADDRESS
00061P [illegible] BD [illegible] A    JSR   EXEPRT  PRINT MESSAGE 3
00062P [illegible] DE [illegible] N    LDX   MSG3P   GET MESSAGE 3 POINTER
00063P [illegible] BD [illegible] A    JSR   EXEPRT  PRINT MESSAGE 3 AGAIN
00064P [illegible] CE [illegible] A    LDX   #MSG4   PRINT MESSAGE 4
00065P [illegible] BD [illegible] A    JSR   EXEPRT
00066                          *
00067                          * MOVE MESSAGE FROM CMSG IN BCOMM2 TO BLANK COMMON
00068                          *
00069P [illegible] CE [illegible]      LDX   #MSGDST MESSAGE DESTINATION ADDRESS
00070P [illegible] FF [illegible]      STX   TOPNTR
00071P [illegible] CE [illegible] N    LDX   #CMSG   MESSAGE ADDRESS (FROM)
00072P [illegible] FF [illegible]      STX   FRMPNT
00073P [illegible]              N      LDAB  CMSGCT  MESSAGE LENGTH

00075P [illegible] FE [illegible] LOOP1 LDX  FRMPNT  GET SOURCE POINTER
00076P [illegible]              A      LDAA  0,X     GET BYTE
00077P [illegible]                     INX           UPDATE SOURCE POINTER
00078P [illegible] FF [illegible]      STX   FRMPNT
00079P [illegible] FE [illegible]      LDX   TOPNTR  GET DESTINATION POINTER
00080P [illegible]              A      STAA  0,X     SAVE BYTE
00081P [illegible]                     INX           UPDATE DESTINATION POINTER
00082P [illegible] FF [illegible]      STX   TOPNTR
00083P [illegible] 5A                  DECB          UPDATE CHARACTER COUNTER
00084P [illegible] 26 EE [illegible]   BNE   LOOP1   LOOP


00086                          *
00087                          * JUMP TO PROGRAM WITH ASCT REGIONS
00088                          *
00089P [illegible] 7E [illegible] A    JMP   ATEST


00091P 004C      0002 A FRMPNT RMB   2          FROM POINTER
00092P 004E      0002 A TOPNTR RMB   2          TO POINTER


00094                          END
TOTAL ERRORS 00000
```

FIGURE C-1

MESSAGE PROGRAM 1 (continued)



```
PAGE  003   PS1    PROGRAM TO PRINT OUT MESSAGES (MAIN)


 R          ATEST  00037*00089
 NB         BCOMM  00009*
 NB         BCOMM2 00019*
 NB 0001 CMSG   00021*00071
 NB 0000 CMSGCT 00020*00073
 P          EXDENT 00038*
 D  F024 EXEPRT 00043 00045*00055 00061 00062 00065
  P 004C PROMPT 00072 00075 00078 00091*
  P 0034 LOOP1  00075*00084
 DP 0000 MSG1   00010 00031*00043
 NB 0000 MSG1F  00010*00054
 DD 0000 MSG2   00011 00027*00043
 NB 0002 MSG2F  00011*
 RB         MSG3   00012 00038*00060
 NB 0004 MSG3P  00012*00052
 RD         MSG4   00013 00038*00064
 NB 0006 MSG4F  00013*
  C 0000 MSGCST 00024*00069
 DP 0015 PGINE  00043 00060*
 R          FORM2  00038*00056
 R          STACK  00039*00052
 DP 0006 START  00043 00053*
  P 004E TOFNTR 00070 00079 00082 00092*
```

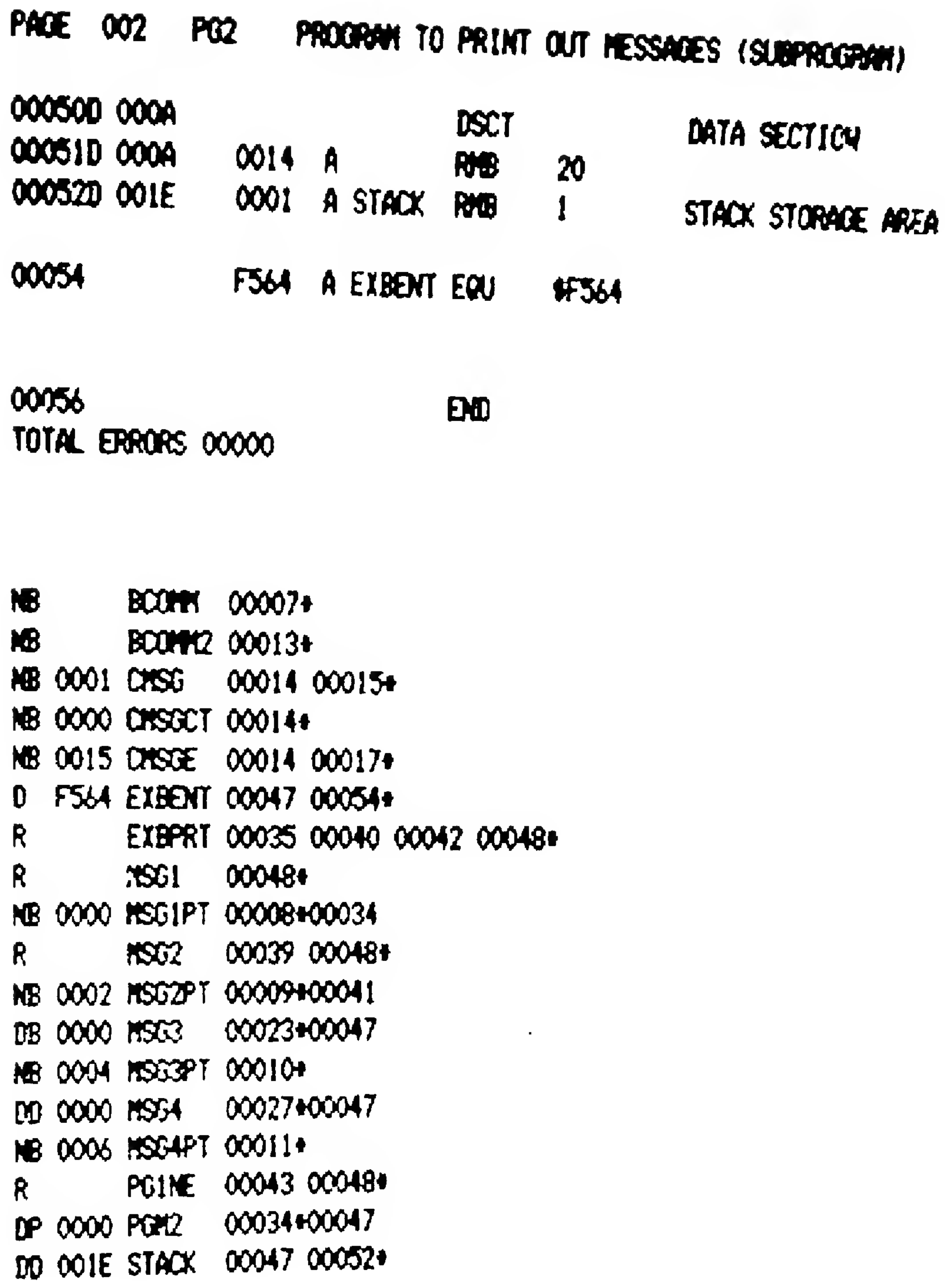
FIGURE C-1

MESSAGE PROGRAM 1 (continued)

```
PAGE  001   PG2    PROGRAM TO PRINT OUT MESSAGES (SUBPROGRAM)

00001                              NAM    PG2
00002                              OPT    CREF,REL,NOG,LLEN=100
00003                              TTL    PROGRAM TO PRINT OUT MESSAGES (SUBPROGRAM)
00004                        *
00005                        * MESSAGE POINTER AREA (BCOMM)
00006                        *
00007N 0000                  BCOMM  COMM   BSCT
00008N 0000     0002  A MSG1PT RMB    2
00009N 0002     0002  A MSG2PT RMB    2
00010N 0004     0002  A MSG3PT RMB    2
00011N 0006     0002  A MSG4PT RMB    2


00013N 0000                  BCOMM2 COMM   BSCT
00014N 0000     14    A CMSGCT FCB    CMSGE-CMSG
00015N 0001     43    A CMSG   FCC    /COMMON TEST PROGRAM/
00016N 0014     04    A        FCB    4
00017           0015  N CMSGE  EQU    *        END OF MESSAGE


00019                        *
00020                        * MESSAGES 3 AND 4
00021                        *
00022B 0000                        BSCT
00023B 0000     4D    A MSG3   FCC    /MESSAGE 3/
00024B 0009     04    A        FCB    4


00026D 0000                        DSCT
00027D 0000     4D    A MSG4   FCC    /MESSAGE 4/
00028D 0009     04    A        FCB    4


00030                        *
00031                        * START OF PROGRAM 2
00032                        *
00033P 0000                        PSCT
00034P 0000 DE 00     N PGM2   LDX    MSG1PT   PRINT MESSAGE 1
00035P 0002 BD 0000   A        JSR    EXBPRT
00036                        *
00037                        * PRINT MESSAGE 2
00038                        *
00039P 0005 CE 0000   A        LDX    #MSG2    PRINT MESSAGE 2
00040P 0008 BD 0000   A        JSR    EXBPRT
00041P 000B DE 02     N        LDX    MSG2PT   PRINT MESSAGE 2 AGAIN
00042P 000D BD 0000   A        JSR    EXBPRT
00043P 0010 7E 0000   A        JMP    PG1NE    RETURN TO PROGRAM ONE
00044                        *
00045                        * XDEFS AND XREFS
00046                        *
00047                              XDEF   MSG3,MSG4,STACK,EXBENT,PGM2
00048                              XREF   EXBPRT,PG1NE,MSG1,MSG2
```

FIGURE C-2

MESSAGE PROGRAM 2



```
PAGE  002   PG2    PROGRAM TO PRINT OUT MESSAGES (SUBPROGRAM)

00050D 000A                    DSCT           DATA SECTION
00051D 000A     0014  A        RMB    20
00052D 001E     0001  A STACK  RMB    1       STACK STORAGE AREA

00054           F564  A EXBENT EQU    $F564


00056                          END
TOTAL ERRORS 00000



NB        BCOMM  00007*
NB        BCOMM2 00013*
NB 0001 CMSG   00014 00015*
NB 0000 CMSGCT 00014*
NB 0015 CMSGE  00014 00017*
D  F564 EXBENT 00047 00054*
R       EXBPRT 00035 00040 00042 00048*
R       MSG1   00048*
NB 0000 MSG1PT 00008*00034
R       MSG2   00039 00048*
NB 0002 MSG2PT 00009*00041
DB 0000 MSG3   00023*00047
NB 0004 MSG3PT 00010*
DD 0000 MSG4   00027*00047
NB 0006 MSG4PT 00011*
R       PG1NE  00043 00048*
DP 0000 PGM2   00034*00047
DD 001E STACK  00047 00052*
```

FIGURE C-2

MESSAGE PROGRAM 2 (continued)

```
PAGE  001   PG3     ***PROGRAM TO ILLUSTRATE USE OF ASCT

00001                          NAM    PG3
00002                          TTL    ***PROGRAM TO ILLUSTRATE USE OF ASCT
00003                          OPT    REL,CREF,LLEN=100


00005                    * BLANK COMMON

00007C 0000                    CSCT
00008C 0000    0020  A CMSG    RMB    $20

00010                          XDEF   ATEST
00011                          XREF   EXBPPT,EXBENT


00013A 0000                    ASCT          UNNECESSARY SINCE ORG CAUSES ASCT ENTRY
00014A 1006                    ORG    $1006
00015A 1006 CE 0000  C ATEST   LDX    #CMSG  START OF COMMON MESSAGE
00016A 1009 7E 1000  A         JMP    ATEST2

00018A 1000                    ORG    $1000
00019A 1000 BD 0000  A ATEST2  JSR    EXBPPT PRINT MESSAGE
00020A 1003 7E 0000  A         JMP    EXBENT GOTO EXBUG-DON'T STOP


00022                          END
TOTAL ERRORS 00000


D  1006 ATEST   00010 00015*
   1000 ATEST2  00016 00019*
 C 0000 CMSG    00008*00015
R       EXBENT  00011*00020
R       EXBPPT  00011*00019
```

FIGURE C-3

MESSAGE PROGRAM 3

```
M6800 LINKING LOADER REV 1.0
?LOAD=PG10,PG20                                  LOAD TWO FILES PG10,PG20
?OI=PG30;LOAD                                    LOAD FILE PG30
?BO=BOTST                                        NAME ABS OBJECT FILE
?ABSP                                            START PASS II
?LOAD=PG10,PG20                                  REPEAT PASS I COMMANDS
?OI=PG30;LOAD
?MAPF                                            PRINT FULL MAP
   NO UNDEFINED SYMBOLS
MAP
  S SIZE  STR  END COMN
  A 0006 1000 1005
  A 0006 1006 100B
  B 0027 0020 0046 001D
  C 0030 0050 007F 0030
  D 0029 009B 00C3 0000
  P 0063 00E6 0148 0000
 MODULE NAME BSCT DSCT PSCT
   PG1         0020 009B 00E6
   PG2         0020 00A5 0136
   PG3         002A 00C4 0149
 COMMON
  NAME   S SIZE  STR
 BCOMM   B 0008 002A
 BCOMM2  B 0015 0032
 DEFINED SYMBOLS
  NAME   S  STR   NAME   S  STR   NAME   S  STR   NAME   S  STR   NAME   S  STR
 ATEST   A 1005   EXBENT A F564   EXBPRT A F024   MSG1   P 00E6   MSG2   D 009B
 MSG3    B 0020   MSG4   D 00A5   PGINE  P 00FB   PGM2   P 0136   STACK  D 00C3
 START   P 00F0
 ?EXIT                                           RETURN TO DISC OPERATING SYSTEM

 !LOAD,BOTST

 EXBUG 1.2 MAID
 *F0;G
 MESSAGE 1
 MESSAGE 1
 MESSAGE 2
 MESSAGE 2
 MESSAGE 3
 MESSAGE 3
 MESSAGE 4
 COMMON TEST PROGRAM
 EXBUG 1.2
```

FIGURE C-4

EXAMPLE OF LOADER WITHOUT INTERMEDIATE FILE

```
M6800 LINKING LOADER REV 1.0
?IF=F1                                  CREATE AN INTERMEDIATE FILE - F1
?STRP=$500;STRD=$400                    ASSIGN STARTING ADDRESS TO PSCT,DSCT
?CURP=$100                              START PSCT MODULES ON ADDRESSES MODULO 100,HEX
?MAPC                                   PRINT USER ASSIGN VALUES
 S SIZE  STR  END  COMN
 B 0000 FFFF 0000 0000
 C 0000 FFFF 0000 0000
 D 0000 0400 0000 0000
 P 0000 0500 0000 0000
?LOAD=PG10,PG20,PG30                     LOAD FILES PG10,PG20,PG30
?BO=B1;ABSP                              ASSIGN ABS OBJECT FILE - B1
 S SIZE  STR  END  COMN
 A 0006 1000 1005
 A 0006 1006 100B
 B 0027 0020 0046 001D
 C 0030 0050 007F 0030
 D 0029 0400 0428 0000
 P 0200 0500 06FF 0000
?MAPF                                    PRINT FULL MAP
  NO UNDEFINED SYMBOLS
MAP
  S SIZE  STR  END  COMN
  A 0006 1000 1005
  A 0006 1006 100B
  B 0027 0020 0046 0010
  C 0030 0050 007F 0030
  D 0029 0400 0428 0000
  F 0200 0500 06FF 0000
 MODULE NAME BSCT DSCT PSCT
   PG1             0020 0400 0500
   PG2             0020 040A 0600
   PG3             002A 0429 0700
 COMMON
  NAME   S  SIZE  STR
 BCOMM   B  0009  002A
 BCOMM2  B  0015  0033
 DEFINED SYMBOLS
  NAME   S  STR    NAME   S  STR    NAME   S  STR    NAME   S  STR    NAME   S  STR
 ATEST   A  1006   EBENT  A  F564   EXBPRT A  F024   MSG1   P  0500   MSG2   D  0400
 MSG3    B  0030   MSG4   D  040A   PGINE  P  0515   PGM2   P  0600   .STACK D  0428
 START   P  050A                         RETURN TO DISC OPERATING SYSTEM
 ?EXIT

 !LOAD,B1

 EXBUG 1.2 MAID
 *50A;G
 MESSAGE 1
 MESSAGE 1
 MESSAGE 2
 MESSAGE 2
 MESSAGE 3
 MESSAGE 3
 MESSAGE 4
 COMMON TEST PROGRAM
 EXBUG 1.2
```

FIGURE C-5

EXAMPLE OF LOADER WITH INTERMEDIATE FILE

```
M6800 LINKING LOADER REV 1.0
?IDON                                      PRINT MODULE INFORMATION
?IF=F3                                     CREATE INTERMEDIATE FILE - F3
?LOAD=PG10                                 LOAD FILE PG10
  PG1       MESSAGE PROGRAM 1
?IFOF                                      TURN 'IF' OFF
?MAPU;IFON                                 PRINT UNDEFINED SYMBOLS, TURN 'IF' ON
 ATEST   EXBENT MSG3    MSG4    PGM2    STACK
0006 UNDEFINED SYMBOLS
?                                          FILE PG120 CONTAINS MODULES PG1 AND PG2
?LIB=PG120:1                               LIBRARY SEARCH OF FILE PG120 ON DRIVE 1
  PG1       MESSAGE PROGRAM 1
  PG2
?IFOF                                      ONLY MODULE PG2 WAS LOADED, TURN 'IF' OFF
?MAPU;IFON                                 LIST UNDEFINED SYMBOLS, TURN 'IF' ON
 ATEST
0001 UNDEFINED SYMBOLS
?LOAD=PG30                                 LOAD MODULE PG3
  PG3
?BO=B3;IDOF;ABSF                           DEFINE OBJECT FILE BO, START PASS II
?MAPF                                      FULL MAP
  NO UNDEFINED SYMBOLS
MAP
 S SIZE  STR  END COMN
 A 0006 1000 1005
 A 0006 1006 100B
 B 0027 0020 0046 001D
 C 0030 0050 007F 0030
 D 0029 009B 00C3 0000
 P 0063 00E6 0148 0000
MODULE NAME BSCT DSCT PSCT
  PG1        0020 009B 00E6
  PG2        0020 00A5 0136
  PG3        002A 00C4 0149
COMMON
 NAME   S SIZE  STR
BCOMM   B 0008 002A
BCOMM2  B 0015 0032
DEFINED SYMBOLS
 NAME   S  STR    NAME   S  STR    NAME   S  STR    NAME   S  STR    NAME   S  STR
ATEST   A 1006   EXBENT  A F564   EXBPRT  A F024   MSG1    P 00E6   MSG2    D 009B
MSG3    B 0020   MSG4    D 00A5   PGINE   P 00FB   PGM2    P 0136   STACK   D 00C3
START   P 00F0
?EXIT                                      RETURN TO DISC OPERATING SYSTEM
```

FIGURE C-6

EXAMPLE OF LIBRARY FILES